Amendments to the Claims:

Following is a complete listing of the claims pending in the application, as amended:

1-29. Cancelled

30. (Currently Amended) A method in a computer system for a server to coordinate assignment of a resource to clients, the method comprising:

assigning the resource to a client;

receiving notification from the client assigned to the resource that the client is waiting for an occurrence of an event before the resource can be productively used;

upon receiving the notification, unassigning the resource from the client;

receiving an indication that the event has occurred; and

after ~~occurrence of the event~~receiving the indication, reassigning the resource to the client.

31. (Original) The method of claim 30 wherein the server upon receiving the notification assigns the resource to another client.

32. (Original) The method of claim 30 wherein the server is an operating system, the clients are tasks, and the resource is a processor resource of the computer system.

33. (Original) The method of claim 32 wherein the processor resource is single-threaded.

34. (Original) The method of claim 32 wherein the processor resource is multi-threaded.

35. (Original) The method of claim 32 wherein the operating system upon notification assigns the processor resource to another task.

36. (Original) The method of claim 32 wherein the task can only perform idle processing until the event occurs.

37. (Original) The method of claim 32 wherein the task can perform no processing until the event occurs.

38. (Original) The method of claim 32 wherein the operating system reassigns the processor resource to the task whenever an external event directed to the task occurs.

39. (Original) The method of claim 32 wherein the notification is received by the operating system in response to informing the task that the task is to be swapped out from processor resource utilization.

40. (Original) The method of claim 32 wherein the operating system assigns a task to the processor resource by assigning the task to a domain.

41. (Original) A method in a computer system for clients to coordinate assignment of a resource by a server, the method comprising:

determining whether the client cannot productively use the resource until an event occurs; and

when the client determines that it cannot productively use the resource, notifying the server that the client cannot productively use the resource

wherein the server unassigns the resource from the client until after an event occurs.

42. (Original) The method of claim 41 wherein the server is an operating system, the clients are tasks, and the resource is a processor resource of the computer system.

43. (Original) The method of claim 42 wherein the processor resource is single-threaded.

44. (Original) The method of claim 42 wherein the processor resource is multi-threaded.

45. (Original) The method of claim 42 wherein the operating system upon notification assigns the processor resource to another task.

46. (Original) The method of claim 42 wherein the task can only perform idle processing until the event occurs.

47. (Original) The method of claim 42 wherein the task determines whether it cannot productively use the processor resource until an event occurs after receiving an indication from the operating system that the task is to be swapped out from processor resource utilization.

48. (Previously Presented) The method of claim 42 wherein the task notifies the operating system after receiving an indication from the operating system that the task is to be swapped out from processor resource utilization.

49. (Original) The method of claim 42 wherein the task can perform no processing until the event occurs.

50. (Original) The method of claim 42 wherein the operating system assigns a task to the processor resource by assigning the task to a domain.

51. (Currently Amended) A computer-readable medium for causing an operating system executing on a computer to coordinate assignment of processor resources to task, by:

assigning a processor resource to a task, the task having productive instructions to execute using the assigned processor resource;

receiving notification from the task assigned to the processor resource that the task is waiting for an occurrence of an event before the task can continue execution of the productive instructions

upon receiving the notification, unassigning the processor resource ~~recourse~~ from the task; and

after occurrence of the event, reassigning the processor resource to the task.

52. (Original) The computer-readable medium of claim 51 wherein the operating system upon receiving the notification assigns the processor resource to another task.

53. (Original) The computer-readable medium of claim 51 wherein the processor is single-threaded.

54. (Original) The computer-readable medium of claim 51 wherein the processor is multi-threaded.

55. (Original) The computer-readable medium of claim 51 wherein the task can only perform idle processing until the event occurs.

56. (Original) The computer-readable medium of claim 51 wherein the task can perform no processing until the event occurs.

57. (Original) The computer-readable medium of claim 51 wherein the operating system reassigns the processor resource to the task after an external event directed to the task occurs.

58. (Original) The computer-readable medium of claim 51 wherein the notification is received by the operating system in response to informing the task that the task is to be swapped out from processor resource utilization.

59. (Original) A computer-readable medium for causing clients executing on a computer system to coordinate assignment of a processor resource with an operating system, by:

under control of a task,

determining whether the task cannot productively use the processor resource until an event occurs; and

when the task determines that it cannot productively use the processor resource, notifying the operating system that the task cannot productively use the resource

wherein the operating system unassigns the resource from the task until after an event occurs.

60. (Original) The computer-readable medium of claim 59 wherein the processor is single-threaded.

61. (Original) The computer-readable medium of claim 59 wherein the processor is multi-threaded.

62. (Original) The computer-readable medium of claim 59 wherein the operating system upon notification assigns the processor resource to another task.

63. (Original) The computer-readable medium of claim 59 wherein the task can only perform idle processing until the event occurs.

64. (Original) The computer-readable medium of claim 59 wherein the task determines whether it cannot productively use the processor resource until an event occurs after receiving an indication from the operating system that the task is to be swapped out from processor utilization.

65. (Original) The computer-readable medium of claim 59 wherein the task notifies the operating system after receiving an indication from the operating system that the task is to be swapped out from processor utilization.

66. (Original) The computer-readable medium of claim 59 wherein the task can perform no processing until the event occurs.

67-104. Cancelled

105. (Previously Presented) The method of claim 39 wherein the task comprises multiple streams and wherein a master stream provides the notification that the task is waiting for an occurrence of an event after all the other streams have quit.

106. (Previously Presented) The method of claim 39 wherein the computer system includes multiple processors, each processor having multiple streams for executing threads of the task, the task having one or more teams of thread, each team representing threads executing on a single processor, and including:

for each team, designating one stream that is executing a thread as a team master stream;

designating one stream that is executing a thread as a task master stream for the task;

for each team master stream, notifying the operating system that the team is ready to be swapped out when each other thread of the team has quit its stream; and

for the task master stream, notifying the operating system that the task is ready to be swapped out when each of the other teams have notified the operating system that that team is ready to be swapped out.

107. (Previously Presented) The method of claim 106 wherein the operating system swaps out the task upon receiving the notification that the task is ready to be swapped out.

108. (Previously Presented) The method of claim 106 wherein each stream stores its own state before quitting the stream or notifying the operating system.

109. (Previously Presented) The method of claim 106 wherein each stream that is not a team master stream quits its stream.

110. (Previously Presented) The method of claim 106 wherein the notifying of the operating system by the task master stream includes indicating whether the task is blocked so that the operating system can defer swapping in the task until an event occurs to unblock the task.

111. (Previously Presented) The method of claim 106 wherein each team master stream notifies the operating system of the number of streams that were executing threads so that the operating system can defer swapping in the task until enough streams are available to execute each of the threads that were executing when the task was swapped out.

112. (Previously Presented) A computer system for coordinating assignment of a processor to tasks, comprising:

means for determining whether a task cannot productively use the processor until an event occurs;

means for notifying an operating system that the task cannot productively use the processor when the task determines that it cannot productively use the resource; and

means for unassigning the task from the processor until after an event occurs.

113. (Previously Presented) The computer system of claim 112 wherein the processor is single-threaded.

114. (Previously Presented) The computer system of claim 112 wherein the processor is multi-threaded.

115. (Previously Presented) The computer system of claim 112 wherein the operating system upon notification assigns the processor to another task.

116. (Previously Presented) The computer system of claim 112 wherein the task determines whether it cannot productively use the processor until an event occurs after receiving an indication from the operating system that the task is to be swapped out from processor utilization.

117. (Previously Presented) The computer system of claim 112 wherein the task notifies the operating system after receiving an indication from the operating system that the task is to be swapped out from processor utilization.

118. (Previously Presented) The computer system of claim 112 wherein the task comprises multiple streams and wherein a master stream provides the notification that the task is waiting for an occurrence of an event after all the other streams have quit.

119. (Previously Presented) The computer system of claim 112 including multiple processors, each processor having multiple streams for executing threads of the task, the task having one or more teams of threads, each team representing threads executing on a single processor, and including:

means for designating one stream that is executing a thread as a team master stream for each team;

means for designating one stream that is executing a thread as a task master stream for the task;

means of each team master for notifying the operating system that the team is ready to be swapped out when each other thread of the team has quit its stream; and

means of the task master stream for notifying the operating system that the task is ready to be swapped when each of the other teams have notified the operating system that that team is ready to be swapped out.

120. (Previously Presented) The computer system of claim 119 wherein the operating system swaps out the task upon receiving the notification that the task is ready to be swapped out.

121. (Previously Presented) The computer system of claim 119 wherein each stream stores its own state before quitting the stream or notifying the operating system.

122. (Previously Presented) The computer system of claim 119 wherein each stream that is not a team master stream quits its stream.

123. (Previously Presented) The computer system of claim 119 wherein the means for notifying the operating system by the task master stream includes means for indicating whether the task is blocked so that the operating system can defer swapping in the task until an event occurs to unblock the task.

124. (Previously Presented) The computer system of claim 119 including means of each team master stream for notifying the operating system of the number of streams that were executing threads so that the operating system can defer swapping in the task until enough streams are available to execute each of the threads that were executing when the task was swapped out.